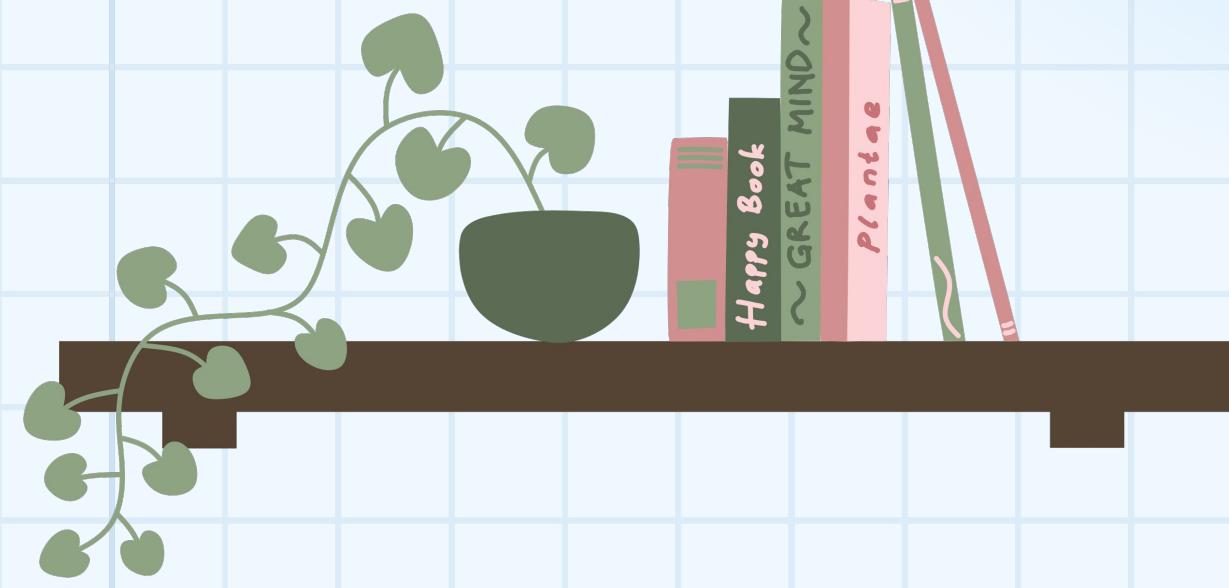
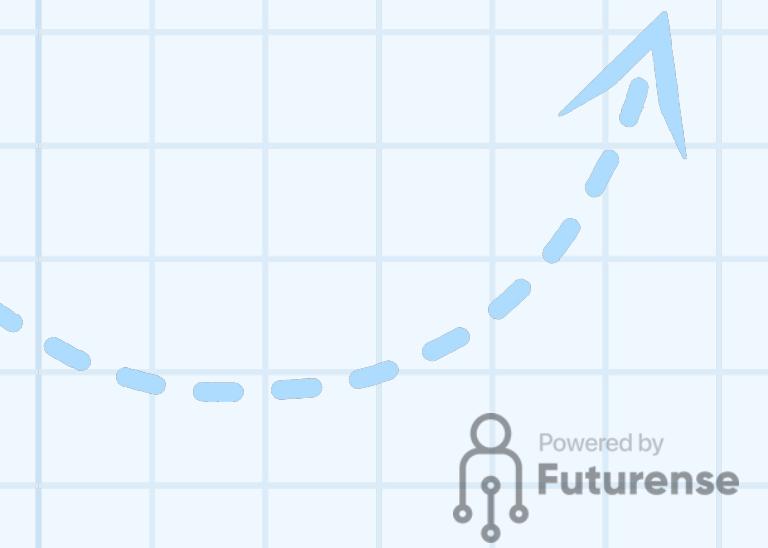
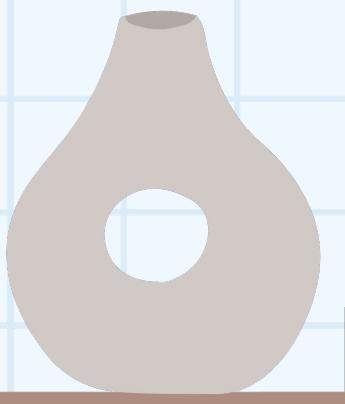
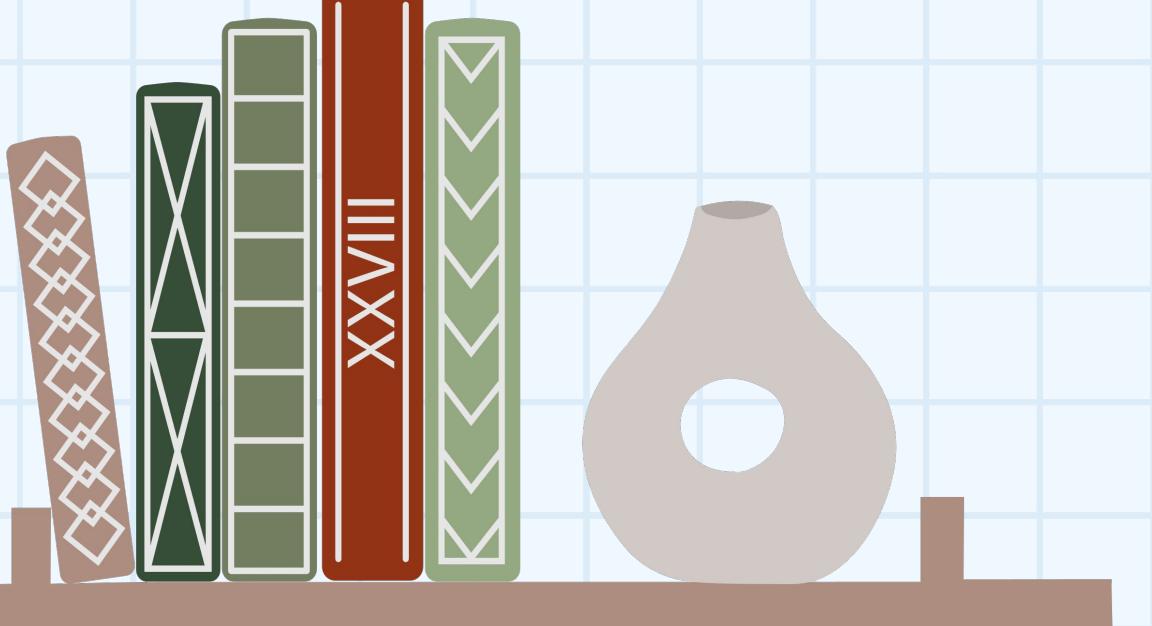




BS./BSC.

Applied AI and Data Science

Algorithmic Thinking & its Applications



Structure vs. Flow

Program Structure Program Flow

- Order code is **presented**
 - Order statements are listed
 - Inside/outside of function
 - Will see other ways...
- Defines possibilities over **multiple executions**
- Order code is **executed**
 - Not the same as structure
 - Some statements duplicated
 - Some statements skipped
- Defines what happens in a **single execution**

Have already seen this difference with functions

Structure vs. Flow: Example

Program Structure Program Flow

```
def foo(): > python foo.py
```

```
    print('Hello')
```

Statement
listed once

```
# Script
```

```
Code foo()
```

```
foo()
```

```
foo()
```

'Hello'
'Hello'
'Hello'

Statement
executed 3x

Bugs occur when flow does
not **match** expectations

Conditionals: If-Statements

Format Example

```
if expression :  
    statement
```

...

```
    statement
```

Indent

```
# Put x in z if it is  
positive if x > 0:
```

```
    z =  
    x
```

Execution:

If *expression* is True, execute all statements **indented** underneath

Python Tutor Example

The screenshot shows the Python Tutor interface. At the top, there is a tab labeled "tab1" with a close button "x" and a plus sign "+" for creating new tabs. Below the tabs is a code editor window containing the following Python code:

```
1 x = 2
2
3 if x > 0
4     print('Hello')
5
6 print('World')
```

Double click the tab to change name, press enter when done.

Visualize

Execute Code

Edit Code

Conditionals: If-Else-Statements

Format Example

```
if expression :  
    statement  
    ...  
else  
    :  
    statement  
    ...
```

```
# Put max of x, y  
in z if x > y:  
    | z = x  
    else  
    : z = y
```

Execution:

If *expression* is True, execute all statements indented under if.

If *expression* is False, execute all statements indented under else.

Python Tutor Example

The screenshot shows a Python Tutor interface. At the top, there is a tab labeled "tab1" with a close button "x" and a plus sign "+" for creating new tabs. Below the tabs is a code editor window containing the following Python code:

```
1 x = 2
2
3 if x > 0
4     print('Hello')
5 else:
6     print('Good-bye')
7
8 print('World')
```

The code uses standard Python syntax, including an if-else statement and print statements. The code editor has a light gray background with white text. The code itself is color-coded: numbers are black, identifiers like "x", "print", and "Hello" are blue, and strings like "Good-bye" and "World" are green.

Double click the tab to change name, press enter when done.

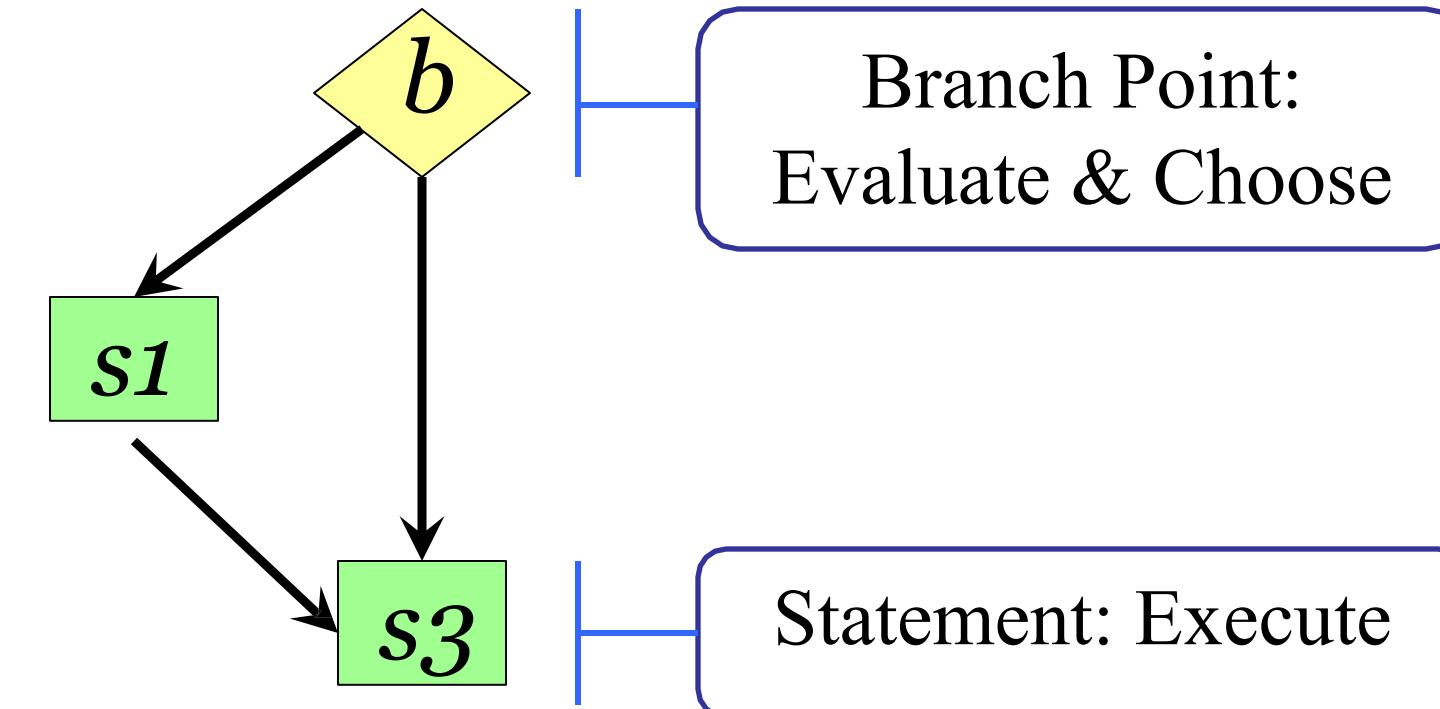
Visualize

Execute Code

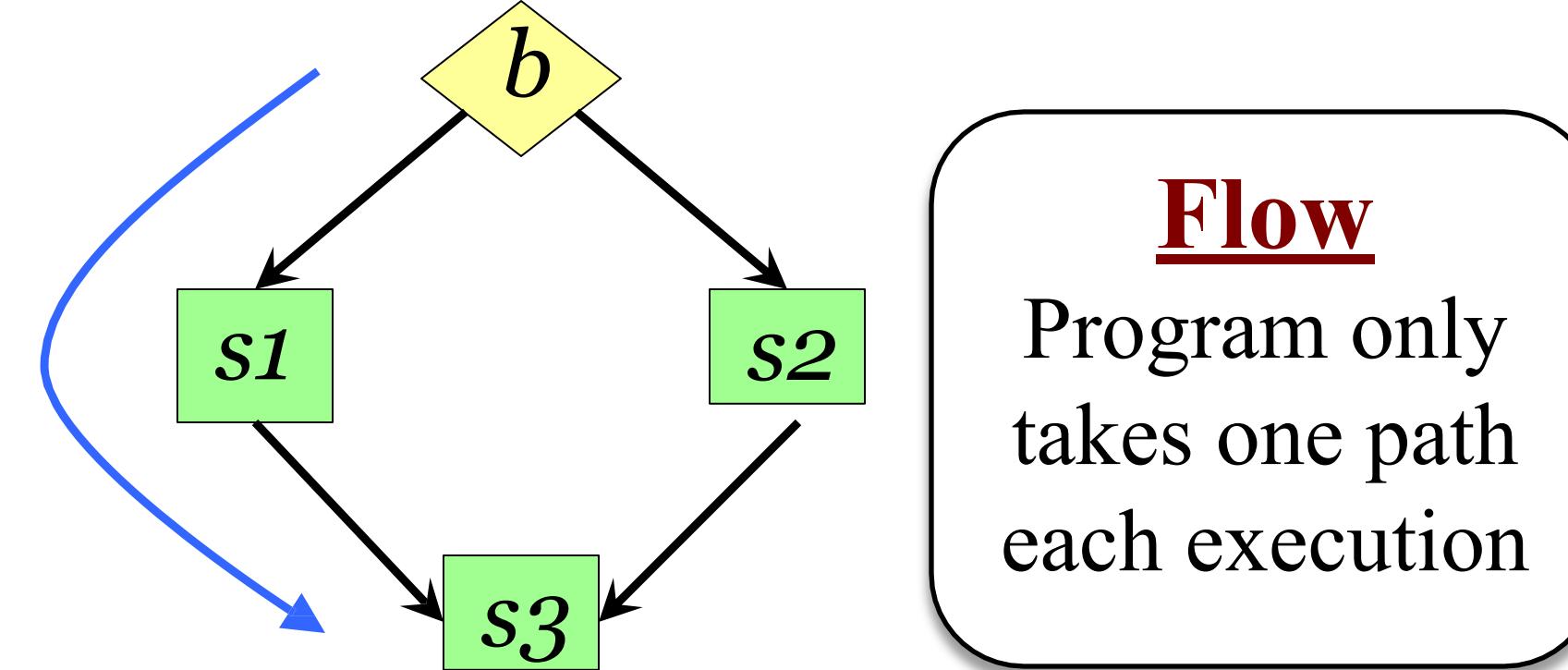
Edit Code

Conditionals: “Control Flow” Statements

```
if b  
:   s1 # statement  
s3
```



```
if b  
:   s1  
else  
:   s2  
s3
```

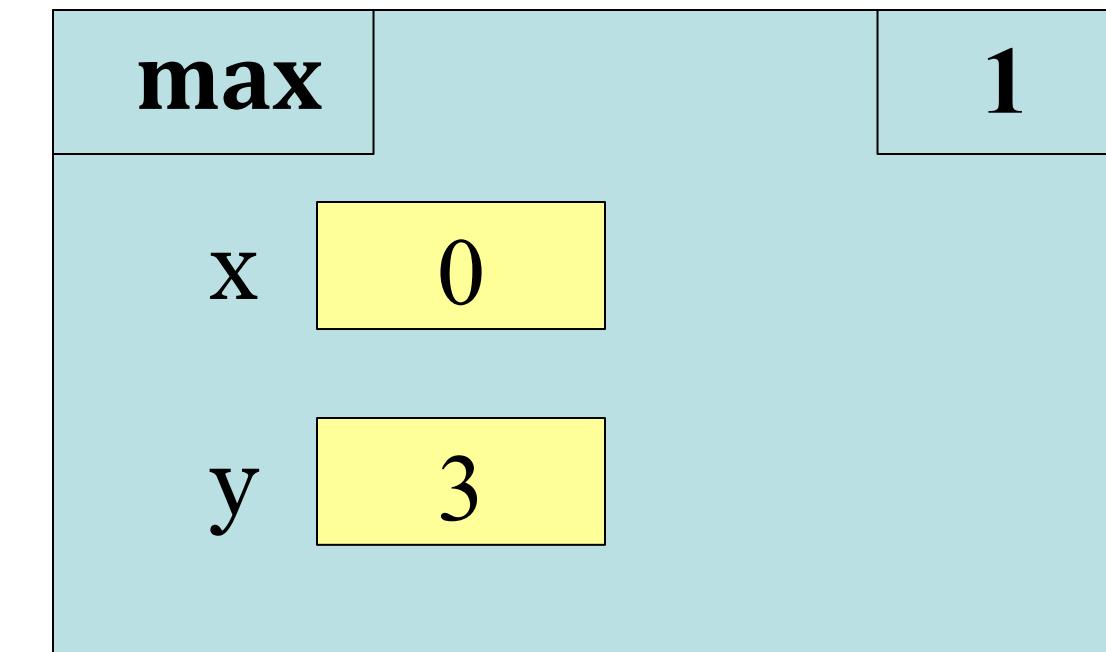


Program Flow and Call Frames

```
def max(x,y):  
    """Returns: max of x,  
    y"""\n    # simple  
    implementation  
  
    1 if x > y:  
        2 return x  
  
    3 return y
```

max(0,3)

:



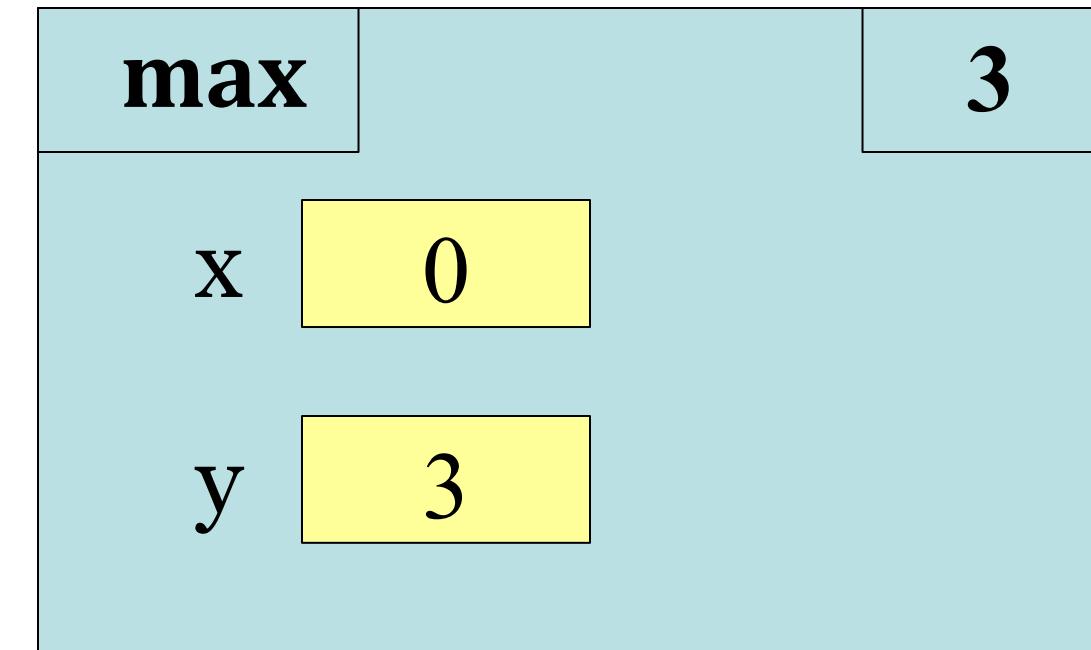
Frame sequence
depends on flow

Program Flow and Call Frames

```
def max(x,y):  
    """Returns: max of x,  
    y"""\n    # simple  
    implementation  
  
    1 if x > y:  
        2 return x  
  
    3 return y
```

Frame sequence
depends on flow

max(0,3):



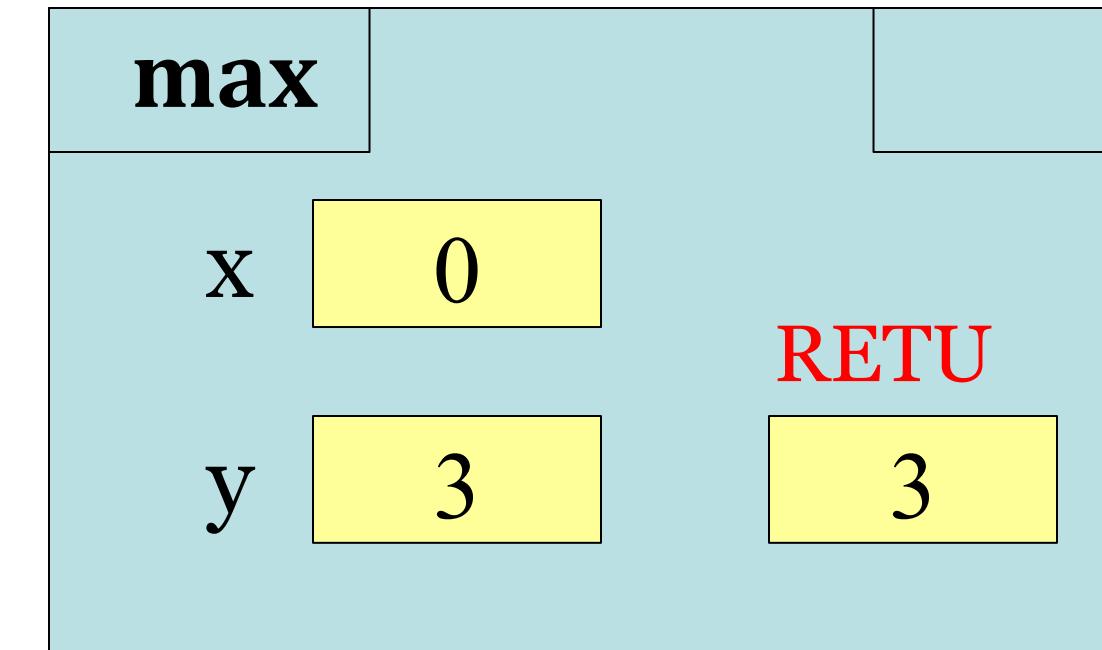
Skips line 2

Program Flow and Call Frames

```
def max(x,y):  
    """Returns: max of x,  
    y"""\n    # simple  
    implementation  
  
1   if x > y:  
2       return x  
  
3   return y
```

Frame sequence
depends on flow

max(0,3):



Skips line 2

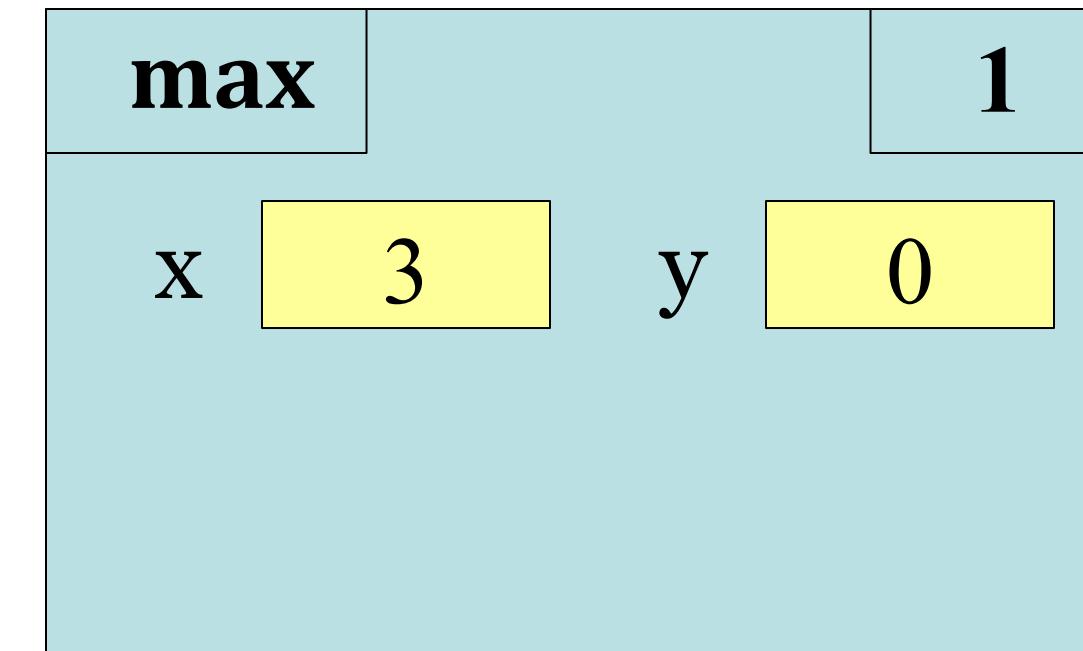
Program Flow vs. Local Variables

```
def max(x,y):
    """Returns: max of x,
    y"""" # swap x, y
    # put the larger in y

1   if x > y:
2       temp = x
3       x = y
4       y = temp

5   return y
```

- max(3,0):



Swaps max
into var y

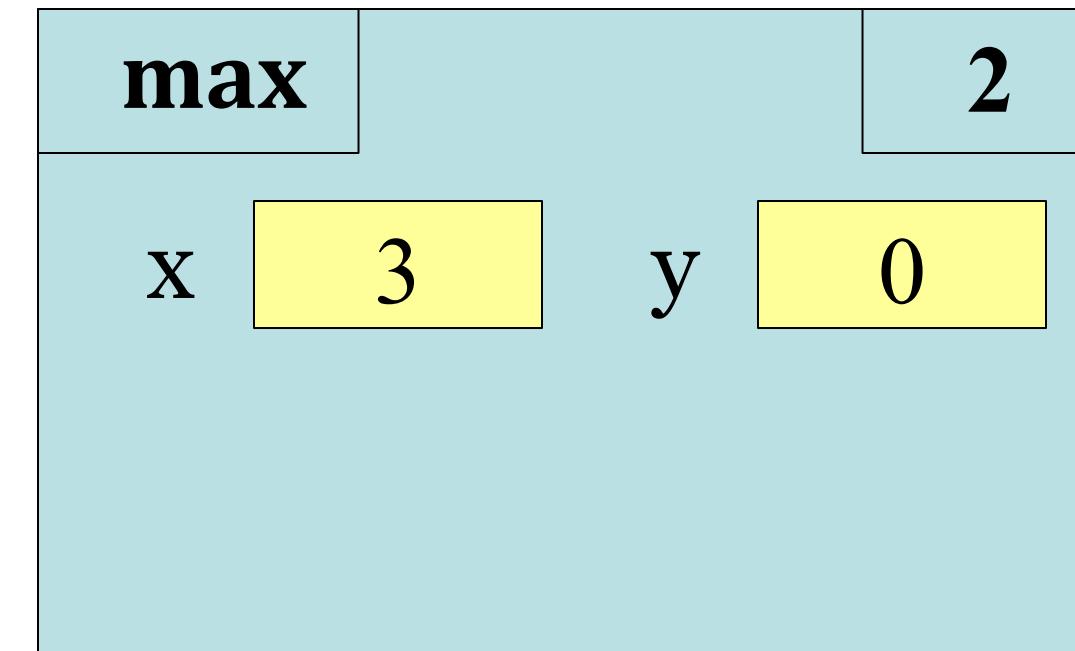
Program Flow vs. Local Variables

```
def max(x,y):
    """Returns: max of x,
    y"""" # swap x, y
    # put the larger in y

1   if x > y:
2       temp = x
3       x = y
4       y = temp

5   return y
```

- max(3,0):



Swaps max
into var y

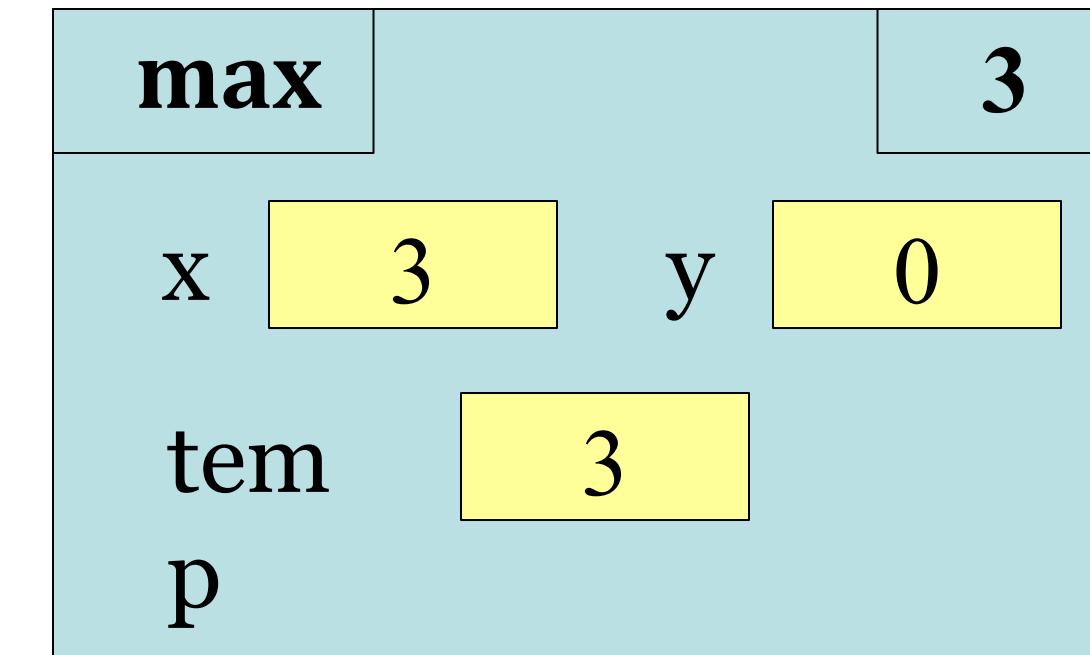
Program Flow vs. Local Variables

```
def max(x,y):
    """Returns: max of x,
    y"""" # swap x, y
    # put the larger in y

1   if x > y:
2       temp = x
3       x = y
4       y = temp

5   return y
```

- max(3,0):



Swaps max
into var y

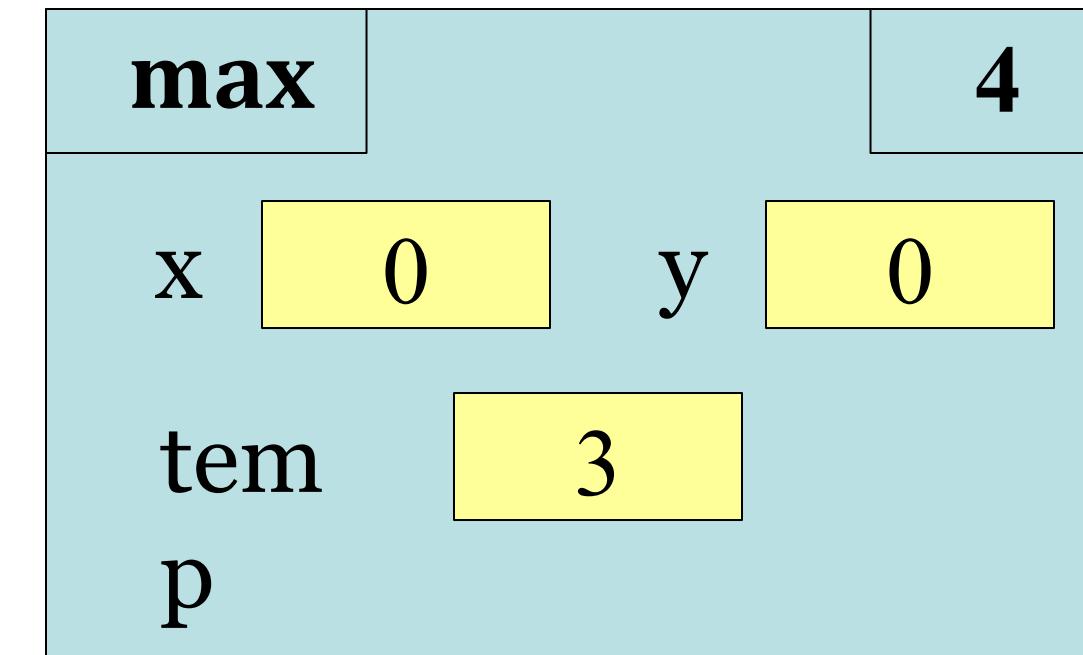
Program Flow vs. Local Variables

```
def max(x,y):
    """Returns: max of x,
    y"""" # swap x, y
    # put the larger in y

1   if x > y:
2       temp = x
3       x = y
4       y = temp

5   return y
```

- max(3,0):



Swaps max
into var y

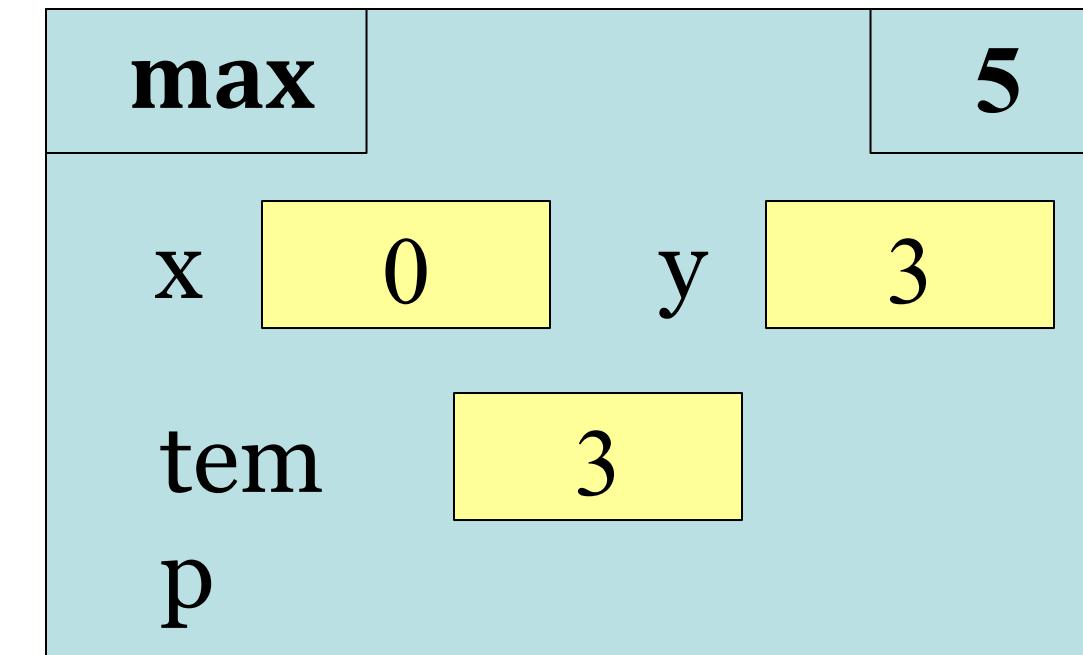
Program Flow vs. Local Variables

```
def max(x,y):
    """Returns: max of x,
    y"""" # swap x, y
    # put the larger in y

1   if x > y:
2       temp = x
3       x = y
4       y = temp

5   return y
```

- max(3,0):



Swaps max
into var y

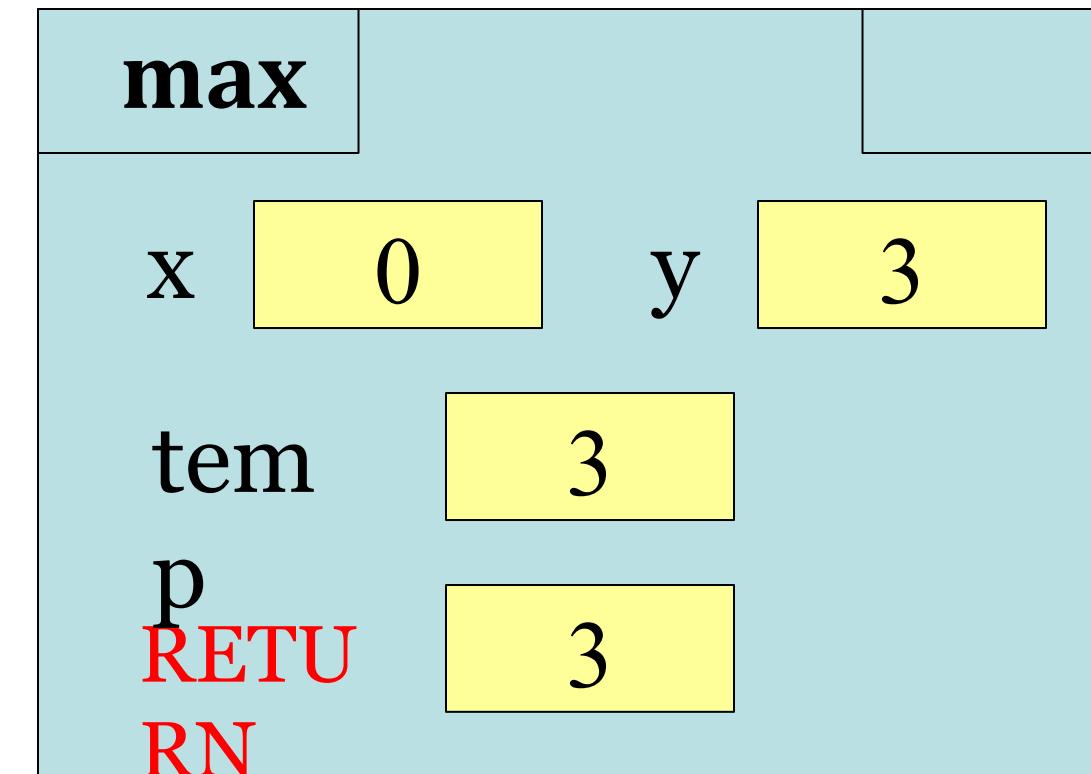
Program Flow vs. Local Variables

```
def max(x,y):
    """Returns: max of x,
    y"""" # swap x, y
    # put the larger in y

1   if x > y:
2       temp = x
3       x = y
4       y = temp

5   return y
```

- max(3,0):



Swaps max
into var y

Program Flow vs. Local Variables

```
def max(x,y):
```

```
    """Returns: max of x, y"""" # swap x, y
```

```
    # put the larger in y
```

```
1   if x > y:
```

```
2       temp = x
```

```
3       x = y
```

```
4       y = temp
```

```
5   return temp
```

- Value of `max(3,0)`?

A: 3

B: 0

C: Error!

D: I do not know

Conditionals: If-Elif-Else-Statements

Format Example

```
if expression :  
    statement
```

...

```
elif expression :  
    statement
```

...

...

```
else
```

```
:    statement
```

...

```
# Put max of x, y, z in w if  
x > y and x > z:
```

```
    w = x
```

```
elif y > z:
```

```
    w = y
```

```
else
```

```
:    w = z
```

Conditionals: If-Elif-Else-Statements

Format Notes on Use

```
if expression :  
    statement  
    ...  
elif expression :  
    statement  
    ...  
...  
else  
    : statement  
    ...
```

- No limit on number of **elif**
 - Can have as many as want
 - Must be between **if**, **else**
- The **else** is always optional
 - **if-elif** by itself is fine
- Booleans checked in order
 - Once it finds first True, skips over all others
 - **else** means **all** are false

Python Tutor Example

tab1 x +

```
1 x = 2
2
3 if x > 0
4     print('Hello')
5 elif x < 0:
6     print('Whatever')
7 else:
8     print('Good-bye')
9
10 print('World')
```

Double click the tab to change name, press enter when done.

Visualize

Execute Code

Edit Code

Conditional Expressions

Format Example

`e1 if bexp else e2`

- `e1` and `e2` are *any* expression
- `bexp` is a boolean expression
- This is an expression!
 - **Evaluates** to `e1` if `bexp` True
 - **Evaluates** to `e2` if `bexp` False

`# Put max of x, y in z z
= x if x > y else y`

expression,
not statement



Thank you

